

# Team K

김일건, 김이지  
정노원, 신경준

# SwithS:

Sungkyunkwan University's  
study integration platform

# Midterm Presentation :

---

**00. Recent feedback**

**01. Objective, Milestone**

**02. Role of each member**

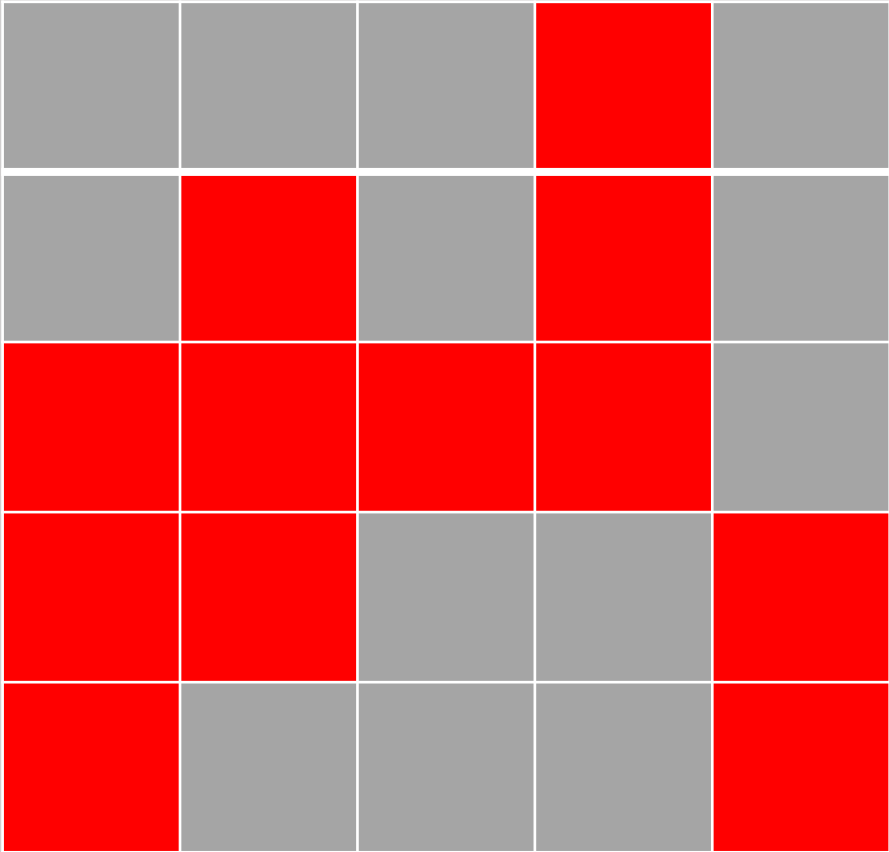
**03. Implementation – Frontend(UI/UX), Backend**

**04. Demo**

**05. Challenges, Limitation**

# 00. Recent feedback

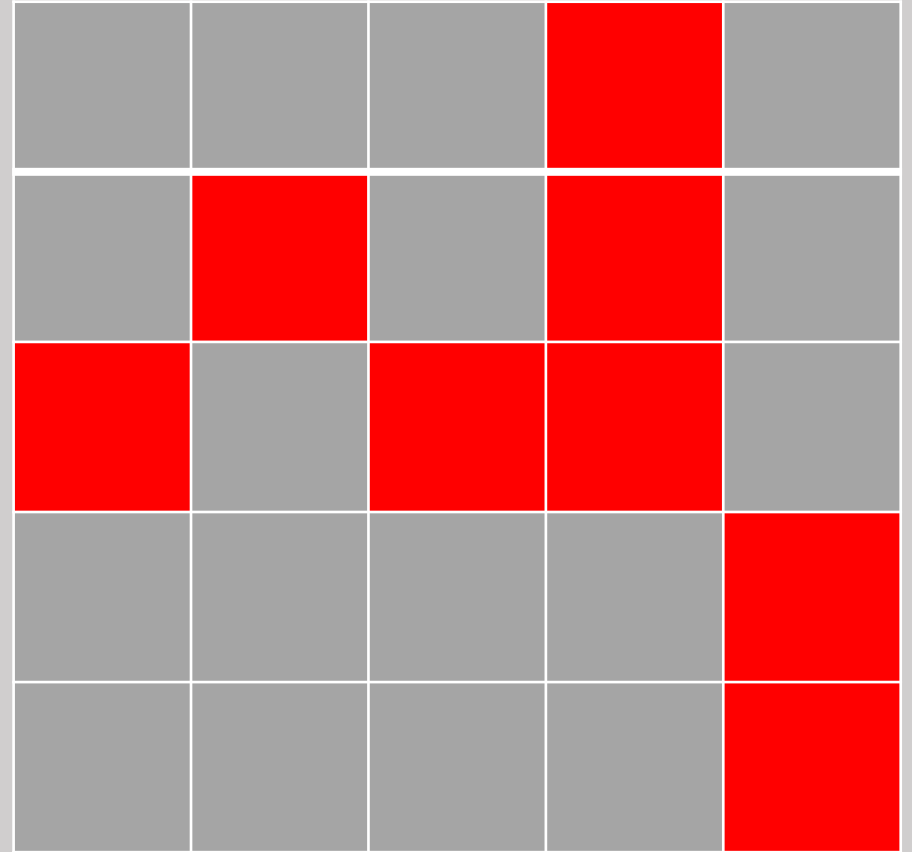
## :00. Recent feedback



Team Member – User A



User B come



Team Member – User A, User B

# 01. Objective, Milestone

# : 01. Objective, Milestone

---

What is the purpose of our project?



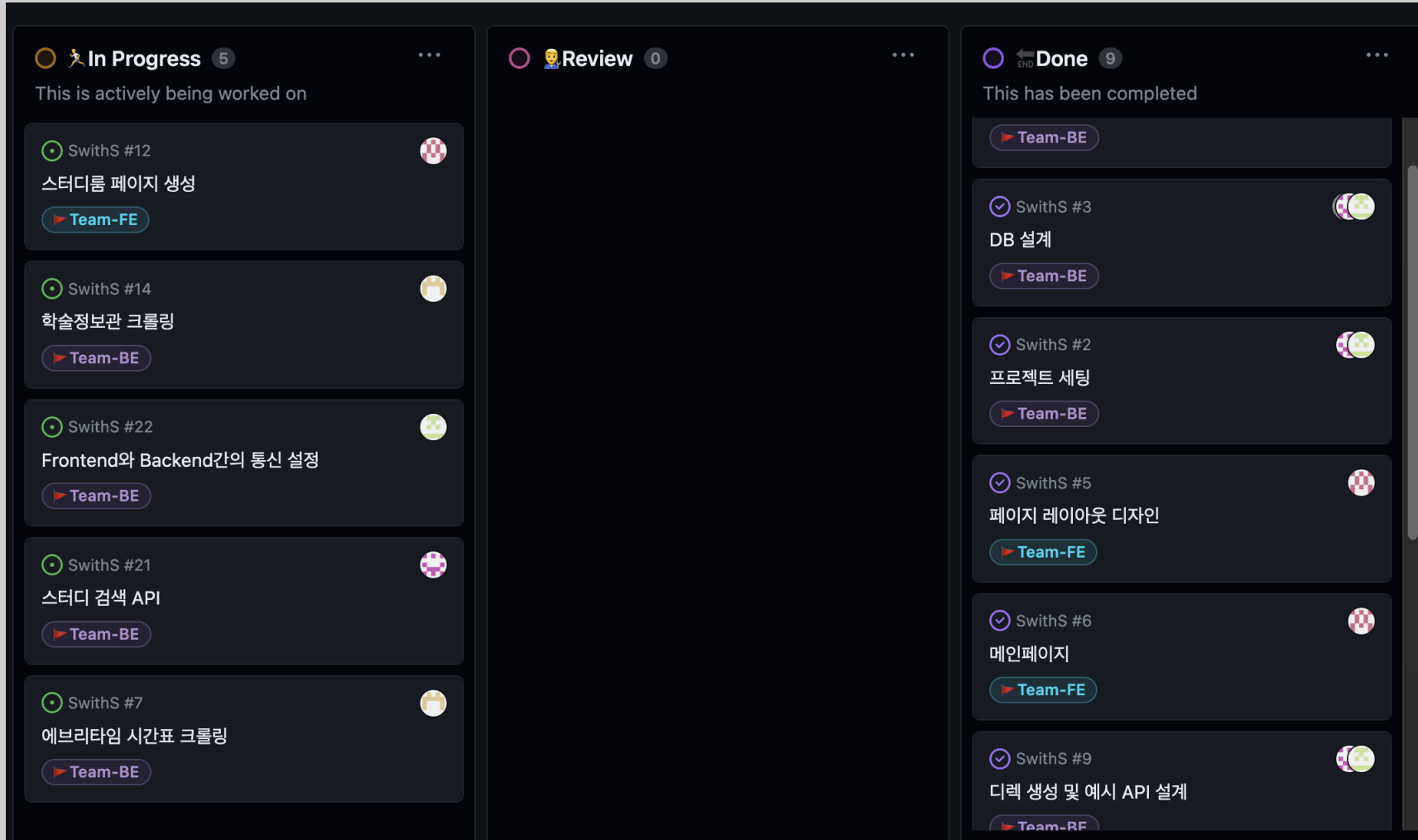
**Study platform for SKKU students!**

## 01. Objective, Milestone

[illegible]

# : 01. Objective, Milestone

- Current





## 02. Role of each member

## • 02. Role of each member

---

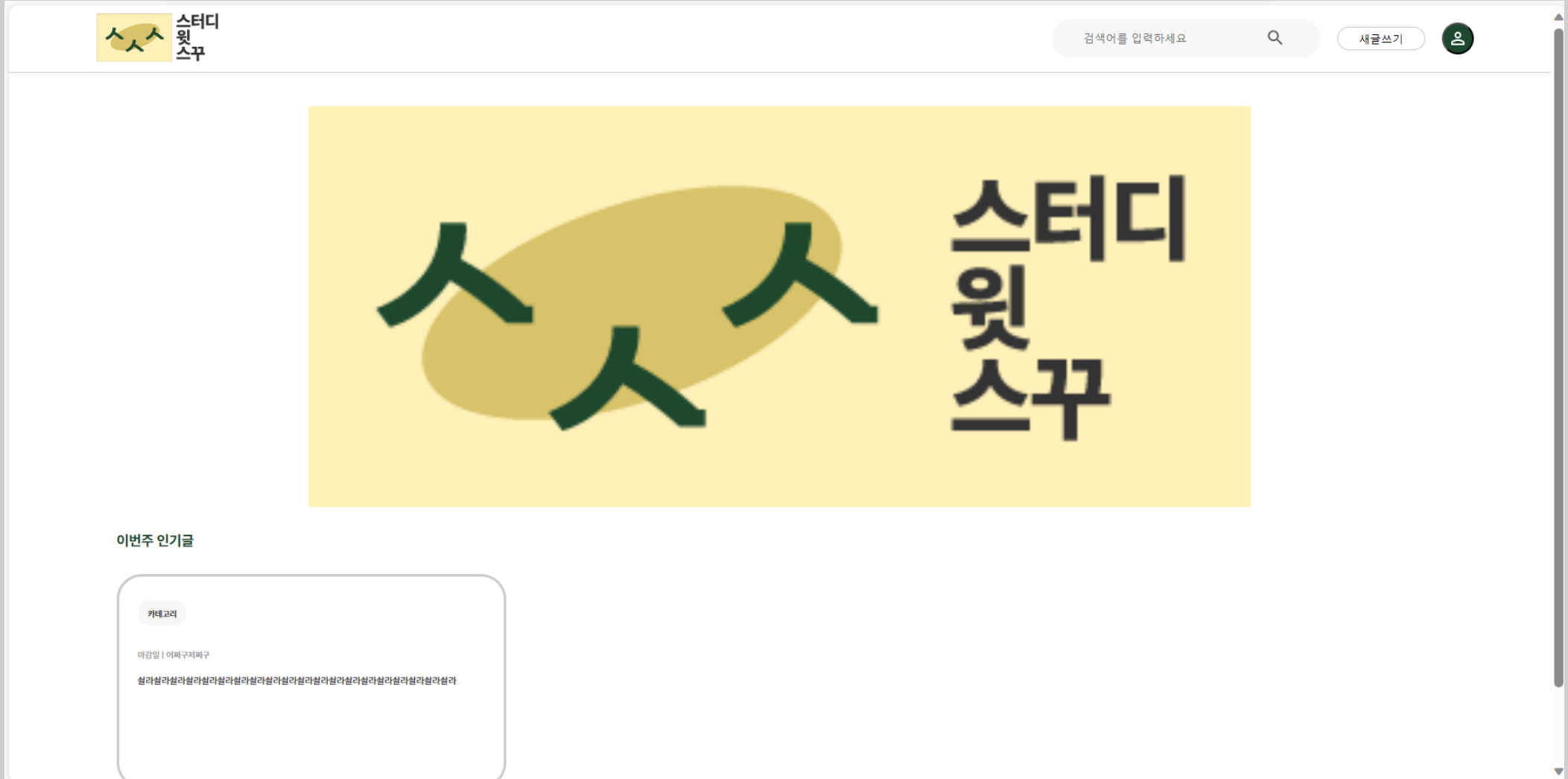
- **김일건** - Project Management, backend API(User, Club) development, Frontend Fix, API Connection
- **김이지** - Frontend page development  
(home, login, register, search, post)
- **신경준** - API statement write, Club API development  
(search get all, findSpecific), entity creation
- **정노원** - DB schema design, Crawling API creation,

# 03. Implementation

- **Frontend(UI/UX)**
- **Backend**

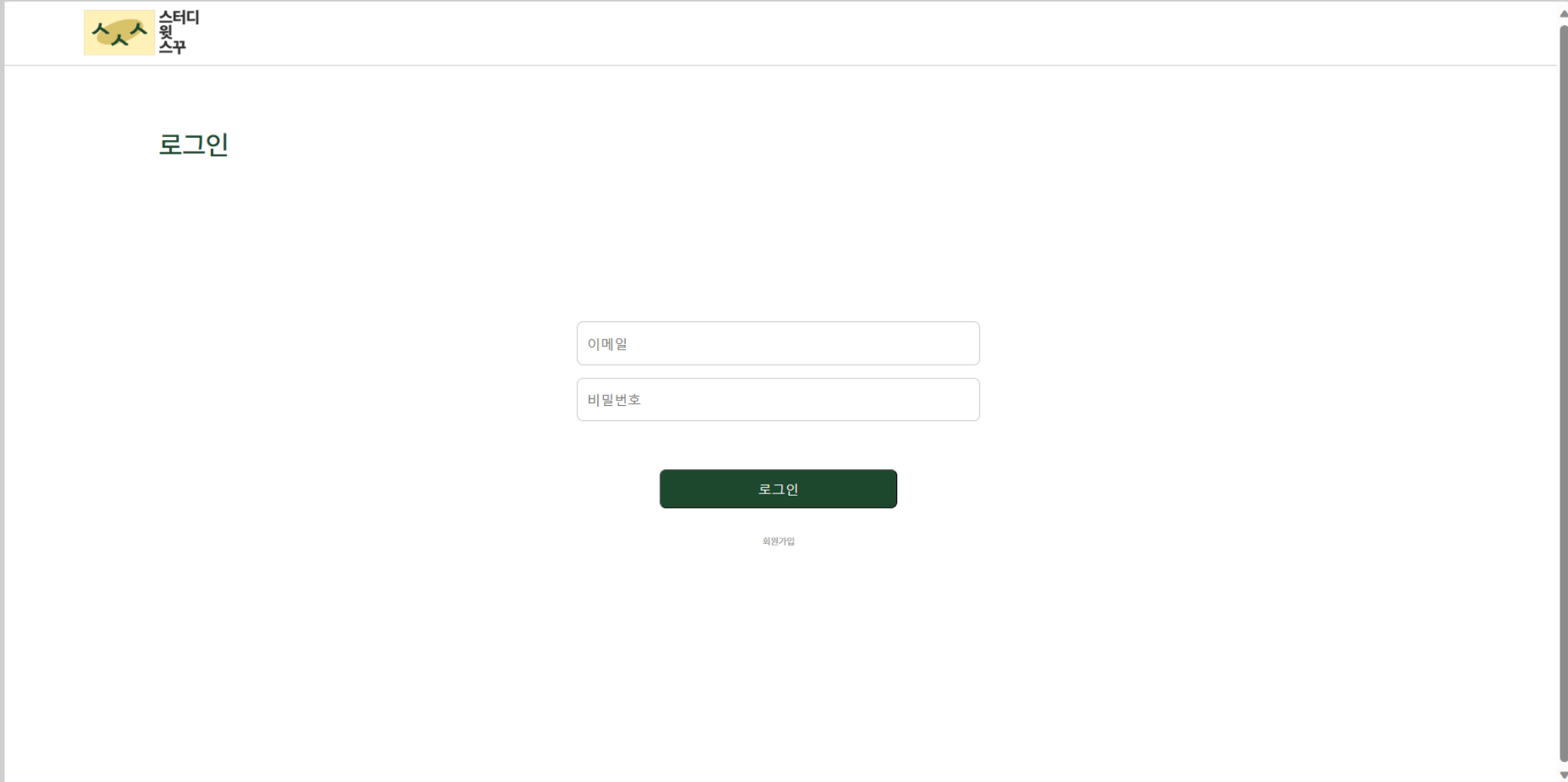
## 03. Implementation – Frontend(UI/UX)

# 1) Home



# : 03. Implementation – Frontend(UI/UX)

## 2) Login



스터디원스꾸

### 로그인

이메일


비밀번호

로그인

[회원가입](#)

# : 03. Implementation – Frontend(UI/UX)

## 3) Register

스터디원스꾸

회원가입

이름

이메일

인증번호

비밀번호


보내기

확인

회원가입

# : 03. Implementation – Frontend(UI/UX)

## 4) Post

 스터디  
랫스꾸

스터디 정보를 입력해주세요

스터디 이름

캡스톤 프로젝트 스터디

스터디 설명

A+를 맞아봅시다 ㅎㅎ

모집구분

November 2023

Su	Mo	Tu	We	Th	Fr	Sa
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2

모집인원

1

모집기간

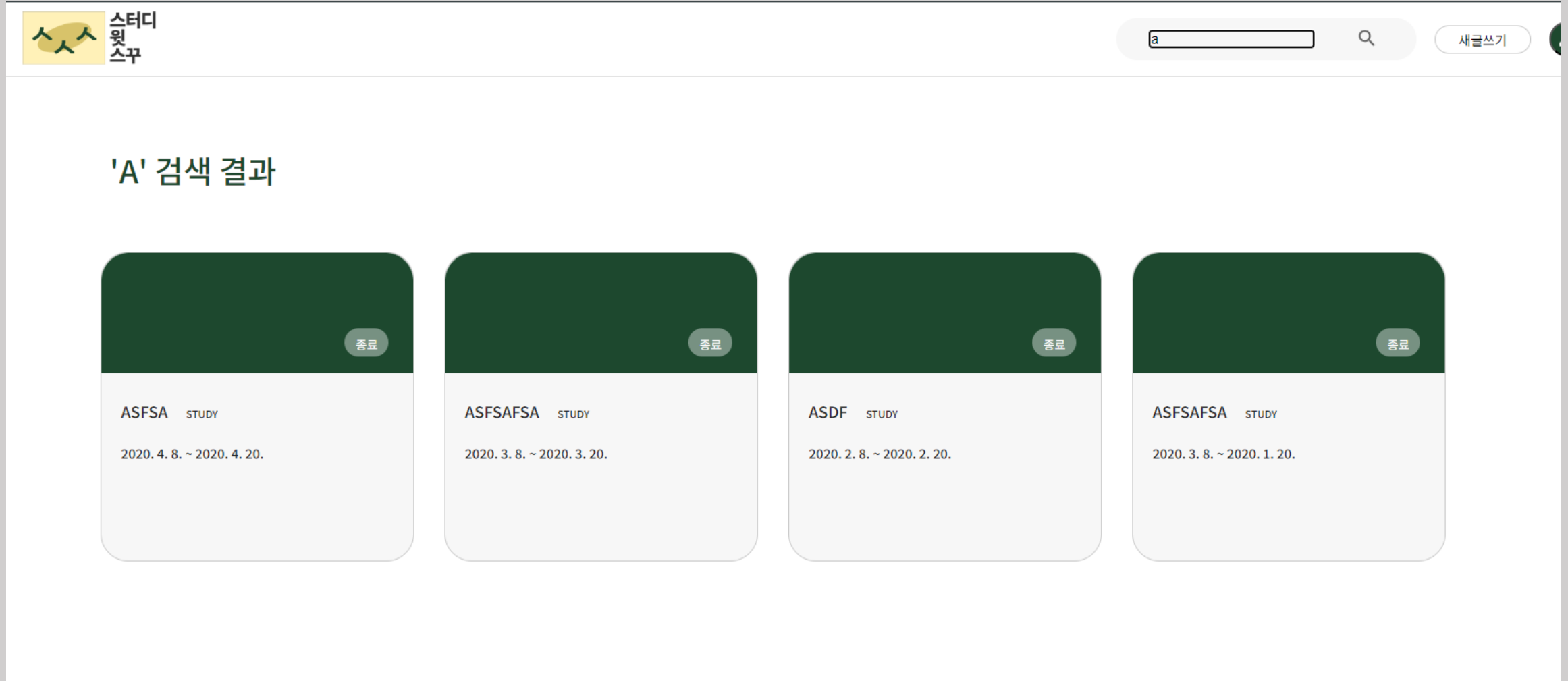
11/09/2023 - 11/24/2023

11/06/2023 -

완료

## : 03. Implementation – Frontend(UI/UX)

### 5) Search





# 03. Implementation – Backend

## 1) DB

```
public class ClubEntity {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    @Column(unique = true, nullable = false)  
    private Long id;  
  
    @Column(nullable = false, length = 20)  
    private String name; // 스터디 이름  
  
    @Column(nullable = false)  
    @Enumerated(EnumType.STRING)  
    private Category category; // 스터디(S) or 멘토링(M)  
  
    @ManyToOne  
    @JoinColumn(name = "major_id")  
    @JsonIgnore // get 요청시 무한참조 현상 발생해서 추가함 TODO: DTO 클래스 처리하는거 생각(이 방법이 더 좋을거 같음)  
    private MajorEntity major; // 전공 스터디인 경우, Major 테이블 참조  
  
    @ManyToOne  
    @JoinColumn(name = "user_id")  
    @JsonIgnore // get 요청시 무한참조 현상 발생해서 추가함 TODO: DTO 클래스 처리하는거 생각(이 방법이 더 좋을거 같음)  
    private UserEntity leader;  
  
    @Column(name = "start_at", nullable = false)  
    private LocalDateTime startAt;  
  
    @Column(name = "end_at", nullable = false)  
    private LocalDateTime endAt;  
  
    @Column(nullable = false, length = 100)  
    private String description;  
  
    @Column(name = "num_recruit", nullable = false)  
    @ColumnDefault("5")  
    private int numRecruit;  
  
}
```

club

```
public class ClubEntity {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    @Column(unique = true, nullable = false)  
    private Long id;  
  
    @Column(nullable = false, length = 20)  
    private String name; // 스터디 이름  
  
    @Column(nullable = false)  
    @Enumerated(EnumType.STRING)  
    private Category category; // 스터디(S) or 멘토링(M)  
  
    @ManyToOne  
    @JoinColumn(name = "major_id")  
    @JsonIgnore // get 요청시 무한참조 현상 발생해서 추가함 TODO: DTO 클래스 처리하는거 생각(이 방법이 더 좋을거 같음)  
    private MajorEntity major; // 전공 스터디인 경우, Major 테이블 참조  
  
    @ManyToOne  
    @JoinColumn(name = "user_id")  
    @JsonIgnore // get 요청시 무한참조 현상 발생해서 추가함 TODO: DTO 클래스 처리하는거 생각(이 방법이 더 좋을거 같음)  
    private UserEntity leader;  
  
    @Column(name = "start_at", nullable = false)  
    private LocalDateTime startAt;  
  
    @Column(name = "end_at", nullable = false)  
    private LocalDateTime endAt;  
  
    @Column(nullable = false, length = 100)  
    private String description;  
  
    @Column(name = "num_recruit", nullable = false)  
    @ColumnDefault("5")  
    private int numRecruit;  
  
}
```

major

## : 03. Implementation – Backend

```
public class VoteEntity {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    @Column(unique = true, nullable = false)  
    private Long id;  
  
    @Column(length = 40, nullable = false)  
    private String title;  
  
    @CreationTimestamp  
    @Column(name = "created_at")  
    private LocalDateTime createdAt;  
  
    @UpdateTimestamp  
    @Column(name = "end_at")  
    private LocalDateTime endAt;  
  
    @OneToMany(mappedBy = "vote")  
    private List<VoteItemEntity> voteItems = new ArrayList<>();  
  
}
```

vote

```
@Table(name = "vote_item_detail")  
public class VoteItemDetailEntity {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    @Column(unique = true, nullable = false)  
    private Long id;  
  
    @ManyToOne  
    @JoinColumn(name = "vote_item_id", nullable = false)  
    private VoteEntity voteItem;  
  
    @ManyToOne  
    @JoinColumn(name = "user_id", nullable = false)  
    private UserEntity user;  
  
}
```

voteItemDetail

## : 03. Implementation – Backend

```
@Table(name = "vote_item")
public class VoteItemEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(unique = true, nullable = false)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "vote_id")
    private VoteEntity vote;

    @Column(name = "start_time")
    private LocalDateTime startTime;

    @Column(name = "end_time")
    private LocalDateTime endTime;
}
```

voteltem

## : 03. Implementation – Backend

```
public class PostEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(unique = true, nullable = false)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "user_id", nullable = false)
    public UserEntity user;

    @Column(length = 20, nullable = false)
    private String title; // 스티디명

    @Column(name = "short_content", length = 20)
    private String shortContent; // 썸네일 상 보이는 간략 설명

    @Column(nullable = false, columnDefinition = "TEXT")
    private String content; // 스티디 설명창을 누르면 보이는 상세 설명 창

    @CreationTimestamp
    @Column(name = "created_at")
    private LocalDateTime createdAt = LocalDateTime.now();

    @UpdateTimestamp
    @Column(name = "updated_at")
    private LocalDateTime updatedAt = LocalDateTime.now();

    @OneToOne
    @JoinColumn(name = "vote_id")
    private VoteEntity vote;
}
```

post

```
public class UserEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(unique = true, nullable = false)
    private Long id;

    @NotBlank(message = "이메일은 필수 입력 값입니다.")
    @Email(message = "이메일 형식에 맞지 않습니다.")
    @Column(length = 30, nullable = false)
    private String email; // 유저 이메일

    @Column(nullable = false)
    @ColumnDefault("false")
    private Boolean admin; // 유저에게 권한을 줄지말지 정함

    @Column(length = 20, nullable = false)
    private String name; // 유저이름

    @Column(length = 100, nullable = false)
    private String password; // 유저 비밀번호

    @ManyToOne
    @JoinColumn(name = "major_id")
    @JsonIgnore
    private MajorEntity major;

    @Column(name = "student_id", length = 10)
    private String studentId; // 학번

    @Column(length = 50)
    private String department; // 학부

    @Column(nullable = false)
    @Enumerated(EnumType.STRING)
    private Statement statement; // 학적상태

    @CreationTimestamp
    @Column(name = "created_at")
    private LocalDateTime createdAt = LocalDateTime.now();

    @UpdateTimestamp
    @Column(name = "updated_at")
    private LocalDateTime updatedAt = LocalDateTime.now();

    @OneToMany(mappedBy = "user")
    private List<PostEntity> posts = new ArrayList<>();

    @OneToMany(mappedBy = "leader")
    private List<ClubEntity> clubLeaderUser = new ArrayList<>();
}
```

user

```
public class UserClubEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(unique = true, nullable = false)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "user_id", nullable = false)
    private UserEntity user;

    @ManyToOne
    @JoinColumn(name = "club_id", nullable = false)
    private ClubEntity club;
}
```

userClub

# 03. Implementation – Backend

## 2) API - club

```
@RestController
@RequiredArgsConstructor
@RequestMapping("/clubs")
public class ClubController {

    private final ClubService clubService;

    // 스터디데이터 전체 조회
    @GetMapping()
    public FindAllClubResponse findAllClub() {
        List<ClubEntity> clubLists = clubService.findAllClub();

        FindAllClubResponse result = new FindAllClubResponse(statusCode:200, isSuccess:true, message:"모든 스터디 조회에 성공하였습니다.", clubLists);
        return result;
    }

    // 스터디 상세 조회
    @GetMapping("/{id}")
    public FindClubResponse findClub(@PathVariable Long id) {
        ClubEntity club = clubService.findClub(id);

        FindClubResponse result = new FindClubResponse(statusCode:200, isSuccess:true, message:"스터디 상세조회에 성공하였습니다.", club);

        return result;
    }

    // 스터디 생성
    @PostMapping()
    public CreateClubResponse createClub(@RequestBody CreateClubRequest createClubRequest) {
        clubService.createClub(createClubRequest);

        CreateClubResponse result = new CreateClubResponse(statusCode:200, isSuccess:true, message:"동아리 생성에 성공하였습니다.");

        return result;
    }

    // 스터디 검색
    @GetMapping("/searchs")
    public FindAllClubResponse searchClub(@RequestParam String keyword) {
        List<ClubEntity> clubLists = clubService.searchClub(keyword);

        FindAllClubResponse result = new FindAllClubResponse(statusCode:200, isSuccess:true, message:"스터디 검색에 성공하였습니다.", clubLists);

        return result;
    }
}
```

[1] Get All Study(GET)

[2] Get Specific Study(GET)

[3] Make Study(POST)

[4] Search Study(GET)

## • 03. Implementation – Backend

### 3) API - user

```
@RestController
@RequiredArgsConstructor
@RequestMapping("/user")
public class UserController {

    private final UserService userService;

    // 유저 조회 API
    @GetMapping()
    public UserEntity getUserById() {
        return userService.getUserById();
    }

    // 유저 생성 API
    @PostMapping()
    @ResponseStatus(HttpStatus.CREATED)
    public CreateUserResponse createUser(@RequestBody CreateUserRequest createUserRequest) {
        userService.createUser(createUserRequest);

        CreateUserResponse result = new CreateUserResponse(code:200, isSuccess:true, msg:"유저 생성에 성공하셨습니다.");

        return result;
    }
}
```

[1] Get User(GET)

[2] Make User(POST)

## 04. Demo

## 05. Challenges, Limitation



## **: 05. Challenges, Limitation**

---

1) Circular reference problem

**[Problem] – Bidirectional Reference between Club Table, Major Table, User Table**

## : 05. Challenges, Limitation

```
@Getter //jackson 오류때문에 entity 바로 반환이 아닌 dto로 변환 후 반환하는 형식으로 코드를 짤.
public static class FindClubResponse {
    private int statusCode;
    private boolean isSuccess;
    private String message;
    private Long id;
    private String name;
    private Category category;
    private String major;
    private Long leader;
    private LocalDateTime startAt;
    private LocalDateTime endAt;
    private String description;
    private int numRecruit;

    public FindClubResponse(int statusCode, boolean isSuccess, String message, ClubEntity entity)
    {
        this.statusCode = statusCode;
        this.isSuccess = isSuccess;
        this.id = entity.getId();
        this.name = entity.getName();
        this.category = entity.getCategory();
        this.major = entity.getMajor().getName();
        this.leader = entity.getLeader().getId();
        this.startAt = entity.getStartAt();
        this.endAt = entity.getEndAt();
        this.description = entity.getDescription();
        this.numRecruit = entity.getNumRecruit();
    }
}
```

**[Trial and Error]**  
**- Use the dto format !!**

## : 05. Challenges, Limitation

```
<name value="해석학"/>
<professor value="김인숙"/>
<time value="화10:30-11:45 【31316】 &lt;br>목09:00-10:15 【31316】 ">
  <data day="1" starttime="126" endtime="141" place="31316"/>
  <data day="3" starttime="108" endtime="123" place="31316"/>
</time>
<place value=""/>
<credit value="3"/>
<closed value="0"/>
</subject>
<subject id="-31489724">
  <internal value=""/>
  <name value="류석현 과외"/>
  <professor value=""/>
  <time value="">
    <data day="1" starttime="228" endtime="252" place=""/>
  </time>
  <place value=""/>
  <credit value="0"/>
  <closed value="0"/>
</subject>
```

**[Problem] – Different format between subject and upload directly**

## : 05. Challenges, Limitation

### [Trial and Error]

- Calculate position of container by size of one of grid

```
List<WebElement> elements = driver.findElements(By.className("grid"));
Dimension gridSize = elements.get(0).getSize();
Point startingPoint = elements.get(0).getLocation();
int startX = startingPoint.x;
int startY = startingPoint.y;
int dx = gridSize.getWidth();
int hour = gridSize.getHeight();
int min30 = hour/2;
int min15 = hour/4; // min15 = 15분 단위

List<WebElement> subjects = driver.findElements(By.className("subject"));
for(WebElement subject : subjects){
    String subjectName = subject.findElement(By.tagName("h3")).getText();

    int currentHeight = subject.getSize().getHeight();
    int minute = currentHeight/min15;
    if(currentHeight % min15 >= min15/2 ) {
        minute += 1;
    }
    minute *= 15;

    int currentX = subject.getLocation().getX();
    int currentY = subject.getLocation().getY();
    currentX = currentX - startX;
    currentY = currentY - startY;
```

```
int day = currentX/dx;
if(currentX % dx >= dx/2 ) {
    day += 1;
}

int startYY = currentY/min30;
if(currentY % min30 > min30/2 ) {
    startYY += 1;
}
LocalTime startTime = LocalTime.of( hour: startYY/2, minute: (startYY%2)*30);
startYY = startYY*30 + minute;

LocalTime endTime = LocalTime.of( hour: startYY/60, minute: startYY%60);
Day currentDay = Day.values()[day];
everytimeComponent component = new everytimeComponent(subjectName,currentDay,startTime,endTime);
System.out.println(component.getName());
result.add(component);
```

## **: 05. Challenges, Limitation**

---

**[Problem] – ID and PASSWORD of SKKU account are required for crawling -> security Problem!**